2.27 DETAILED RESULTS FOR ANGLE TRACKING

The angle servo response has been implemented in *RADGUNS* v.1.8 in subroutines MOVANT, TIMDON, PHIDEL, and matrix utility routines. This code was assessed by comparing it with the requirements and design described in Section 7.1 of the PD[3]. Minor discrepancies were discovered, but fundamentally the FE was correctly implemented as specified.

Table 2.27-1 summarizes the desk checking and software testing verification results for each design element in the Angle Tracking FE. One entry is listed for each of the eight design elements. The results columns contain check marks if no discrepancies were found during verification. Where discrepancies were found, the Desk Check Result column contains references (D1, D2, …) to discrepancies listed in Table 2.27-3, while the Test Case Result column identifies the relevant test cases which explain the discrepancies.

TABLE 2.27-1. Verification Matrix for Angle Tracking FE.

Design Element	Code Location	Desk Check Result	Test Case ID	Test Case Result
27-1: Antenna Response	MOVANT, 189-219	D1	27-1 27-2 27-4 27-5 27-10 27-11 27-12	Y
27-2: Taylor Series	PHIDEL and Matrix Utilities	D2, D3	27-6 27-11	Y
27-3: Clamping of Error Signals	MOVANT, 190-196	D4	27-15	Y
27-4: Scaling Factor	MOVANT, 197-198	Y	27-11	Y
27-5: Derivatives	MOVANT, 199-200	Y	27-11	Y
27-6: Slew Rate Limits	MOVANT, 210-212	D5	27-3	Y
27-7: Manual and Memory Modes	MOVANT, 217-223	Y	27-8 27-9 27-13 27-14	D6
27-8: Track Restoration Logic	MOVANT, 173-186	Y	27-7	Y

2.27.1 Overview

The developer of *RADGUNS* has defined functional element 7.1 Angle Tracking to include only the servo responses. The angle discriminators will be included at a later date. The azimuth and elevation servos move the antenna based on error signals from the discriminators. The ability to null out these error signals is limited by lag and overshoot, gimbal limits, rate limits, and dead zones.

2.27.2 Design Elements

Eight verification design elements are required for angle tracking and are described in Table 2.27-2. Both the azimuth and elevation channels have identical requirements, and both are implemented in parallel in subroutine MOVANT and the routines called by MOVANT.

TABLE 2.27-2. Angle Tracking Design Elements.

Module	Design Element	Description
MOVANT	27-1: Antenna Response	Compute discrete time solution to servo response.
TIMDOM and Matrix Utilities	27-2: Taylor Series	Compute estimates of state matrices.
MOVANT	27-3: Clamping of Error Signals	Apply limits to servo inputs due to motor/clutch responses.
MOVANT	27-4: Scaling Factor	Convert from error signal (V) to angle (rad).
MOVANT	27-5: Derivatives	Compute numerical derivatives.
MOVANT	27-6: Slew Rate Limits	Impose maximum angle rates.
MOVANT	27-7: Manual and Memory Modes	Set pointing angles to command values.
MOVANT	27-8: Track Restoration Logic	Reset working vectors to zero.

2.27.3 Desk Check Activities and Results

The code implementing this FE was manually examined using the procedures described in Section 1.1 of this report. Table 2.27-3 summarizes desk checking verification results for the angle tracking functional element.

TABLE 2.27-3. Angle Tracking Desk Checking Verification Matrix.

Design Element	Desk Check Result
27-1: Antenna Response	D1: The *RADGUNS* manuals have no analytical description of the tracking circuits. There is a programmer description of the MOVANT subroutine which is accurate but incomplete. The branches to treat coast and manual modes are not described, and slew-rate limiting is not covered.
27-1: Antenna Response 27-2: Taylor Series	D2: Most of the FORTRAN arrays in the matrix utility routines are hard-wired to a main index of 4 but the actual maximum index is passed as an argument, possibly creating array out-of-bounds problems.
27-2: Taylor Series	D3: Variable L implicitly declared as an integer but used as a real.
27-3: Clamping or Error Signals	D4: The motor/clutch response is modeled as an amplifier-type gain term between the angle discriminator output (in volts) and the servo input (radians). The implementation is a linearization of Figure E-4 in Reference e. The output in the referenced figure is labeled “deg/s^2” indicating a command acceleration, while in the code the output from this motor/clutch response is loaded into the first position of the angle drive state vector, which corresponds to angle. If the driving function is truly an angular acceleration, it should be loaded into the third position. The figure may be labeled incorrectly. The scaling factors, KAZ and KEL, may convert from acceleration to angle space by multiplying by the square of the time step or the code may be incorrect.
27-6: Slew Rate Limits	D5: The implementation of the slew-rate limits works, but the second derivative is not adjusted to be mathematically consistent.

Internal documentation and code quality problems are characterized in Table 2.27-4.

TABLE 2.27-4. Code Quality and Internal Documentation Results.

Module	Code Quality	Internal Documentation
MOVANT, PHIDEL and Matrix Utilities	The inputs and outputs were examined and found to match Tables 7.1-2, -3, and -4 in the PD[3]. The call hierarchy (Figure 7.1-2 in the PD[3]) and the flow logic (Figure 7.1-3 in the PD[3]) were found to be accurate representation of the FE implementation. The overall code quality in MOVANT and the Utility routines is excellent.	Internal documentation was accurate and generally adequate, although no units are given in variable definitions and a few variables are not defined. No additional assumptions and limitations, beyond those given in the PD[3] (Section 7.1-5), were found.

2.27.4 Software Test Cases and Results

Table 2.27-5 contains a description of the software tests performed for angle tracking. Tests 27-1 through 27-11 were performed using an off-line driver to simulate the discriminator response and the current boresight conditions. In these off-line driver tests, the time step was set to the system CONSCAN period. In all test cases, standard *RADGUNS* system data were used as inputs.

TABLE 2.27-5. Angle Tracking Software Test Cases.

Test Case ID	Test Case Description
27-1	OBJECTIVE: Check that no error results in any movement (zero boresight). PROCEDURE: 1. Set AZERR, ELERR, BSAZ, and BSEL to 0.0. 2. Run code for 1 second of simulation time. 3. Verify that BSAZ and BSEL remain 0.0 at every time step.
27-2	OBJECTIVE: Check that no error results in any movement (non-zero boresight). PROCEDURE: 1. Set AZERR and ELERR TO 0.0, BSAZ to 0.1, and BSEL to 0.0. 2. Run code for 7 seconds of simulation time. 3. Verify that BSAZ remains at 0.1 at every time step.
27-3	OBJECTIVE: Check slew rate limits. PROCEDURE: 1. Set AZERR = 1. 2. Run code for 5 seconds of simulation time. 3. Verify that BSAZ change is limited by maximum slew rate.
27-4	OBJECTIVE: Check that azimuth error is not coupled into elevation channel. PROCEDURE: 1. Set AZERR =1, ELERR = 0, BSEL = 0, BSAZ = 0. 2. Run code for 5 seconds of simulation time. 3. Verify that BSEL = 0.0 at every time step.

TABLE 2.27-5. Angle Tracking Software Test Cases. (Contd.)

Test Case ID	Test Case Description
27-5	OBJECTIVE: Check reasonableness of response to impulse input. PROCEDURE: 1. Use the following values Time step — AZERR — ELERR 0 — 0 — 0 1 — 2 — 0 2 - end — 0 — 0 2. Set BSAZ = 0, BSEL = 0. 3. Run for 3 seconds of simulation time. 4. Examine values of BSAZ at each time step. 5. Verify that the servo lags and then drifts; i.e., that BSAZ increases rapidly for one time step, then increases slowly for the remaining time.
27-6	OBJECTIVE: Check initialization logic. PROCEDURE: 1. Set FIRST = .TRUE. 2. Stop in Subroutine MOVANT. 3. Examine values in matrices [A] and [B] and compare to referenced values. 4. Verify that Subroutine PHIDEL is called.
27-7	OBJECTIVE: Check track restoration logic. PROCEDURE: 1. Set RSTTRK = .FALSE. 2. At third time step, set RSTTRK = TRUE. 3. Examine values of DRIVE and WORK vectors and verify that they are zero.
27-8	OBJECTIVE: Check Memory Mode logical access. PROCEDURE: 1. Set MEMODE = .TRUE. 2. Run 1 time step. 3. Verify that lines 235-240 are accessed. RESULT: For these tests, the derivative of the azimuth position goes to a very large number (curs even with AZERR = 0 (no driving error). This problem was found to occur whenever AZDIFF returned a negative value. The subroutine BOUND then converts the negative value to a large positive value (close to 2 calculation yields a negative number with large absolute value. Another potential problem is that AZLAST is not initialized to AZWORK prior to entering MANAIM or MEMODE branches. This caused no discernible impact on overall tracking since the actual errors during memory and lost track are much worse than could be caused by single point errors in the rates.

TABLE 2.27-5. Angle Tracking Software Test Cases. (Contd.)

<table>
<tr><th>Test Case ID</th><th>Test Case Description</th></tr>
<tr><td>27-9</td><td>OBJECTIVE: Check Manual Mode logical access.
PROCEDURE:
1. Set MANAIM = .TRUE.
2. Run 1 time step.
3. Verify that lines 235-240 are accessed.
RESULT:
For these tests, the derivative of the azimuth position goes to a very large number (curs even with AZERR = 0 (no driving error). This problem was found to occur whenever AZDIFF returned a negative value. The subroutine BOUND then converts the negative value to a large positive value (close to 2 calculation yields a negative number with large absolute value. Another potential problem is that AZLAST is not initialized to AZWORK prior to entering MANAIM or MEMODE branches. This caused no discernible impact on overall tracking since the actual errors during memory and lost track are much worse than could be caused by single point errors in the rates.</td></tr>
<tr><td>27-10</td><td>OBJECTIVE: Check reasonableness of response for oscillating input.
PROCEDURE:
1. Use the following values

Time AZERR ELERR
0.0 1 0
0.2 0 0
1.0 –1 0
1.2 0 0
2.0 +1 0
2.2 0 0

2. Run for 3 seconds of simulation time.
3. Examine value of BSAZ at every time step and check for reasonableness; i.e., that it changes in the correct direction and that it changes rapidly when AZERR is nonzero and changes slowly when AZERR is 0.</td></tr>
<tr><td>27-11</td><td>OBJECTIVE: Check matrix and vector calculations.
PROCEDURE:
1. Set AZERR = 1, ELERR = 0, BSAZ = 0, BSEL = 0.
2. Run for 1 time step.
3. Examine values of matrices [A],[B],[Φ], and [ ], and of vector AZDRIV and AZWORK.
4. Compare to pre-calculated values.</td></tr>
<tr><td>27-12</td><td>OBJECTIVE: Check that the time domain servo response calculations are performed during full-up simulation.
PROCEDURE:
1. Run the full-up RADGUNS Model, using a SINUSOIDAL target flight path with initial position (X = 5000 m, Y = 300 m, Z = 100 m); i.e., a flyby with 0.3 km offset.
2. Use a sinusoidal path with amplitude 10 m and (sine) wavelength of 30 m.
3. Examine the calculated values of BSAZ and BSEL for reasonable response to target path.</td></tr>
</table>

TABLE 2.27-5. Angle Tracking Software Test Cases. (Contd.)

Test Case ID	Test Case Description
27-13	OBJECTIVE: Check that the Memory Mode is invoked when target is masked for short time. PROCEDURE: 1. Run *RADGUNS* with same flight path as test 7.1-12. 2. Input a HILL for $-20 \le x$ 20. This will mask target for about 0.8 s. 3. Verify that MEMORY mode is invoked in MOVANT when target is hidden by hill. RESULT: For these tests, the derivative of the azimuth position goes to a very large number (curs even with AZERR = 0 (no driving error). This problem was found to occur whenever AZDIFF returned a negative value. The subroutine BOUND then converts the negative value to a large positive value (close to 2 calculation yields a negative number with large absolute value. Another potential problem is that AZLAST is not initialized to AZWORK prior to entering MANAIM or MEMODE branches. This caused no discernible impact on overall tracking since the actual errors during memory and lost track are much worse than could be caused by single point errors in the rates.
27-14	OBJECTIVE: Check that the Manual Mode is invoked during break lock. PROCEDURE: 1. Run *RADGUNS* with same flight path as in test 7.1-12. 2. Input a JAMMER into the scenario. 3. Verify that Manual Mode is entered during break lock, and that the Restore Track branch is accessed after tracking is restored. RESULT: For these tests, the derivative of the azimuth position goes to a very large number (curs even with AZERR = 0 (no driving error). This problem was found to occur whenever AZDIFF returned a negative value. The subroutine BOUND then converts the negative value to a large positive value (close to 2 calculation yields a negative number with large absolute value. Another potential problem is that AZLAST is not initialized to AZWORK prior to entering MANAIM or MEMODE branches. This caused no discernible impact on overall tracking since the actual errors during memory and lost track are much worse than could be caused by single point errors in the rates.
27-15	OBJECTIVE: Check Clamping Response. PROCEDURE: 1. Use off-line driver with initial conditions of AZERR = 1, ELERR = 0, BSAZ = 0, BSEL = 0. 2. After first time step set AZERR = 5.0 3. Stop at line 195. 4. Examine AZERR and compare to system clamping limit.

2.27.5 Conclusions and Recommendations

2.27.5.1 Code Discrepancies

The code for the angle tracking FE is fundamentally a correct implementation of the design specified in the PD[3]. However, the reader should refer to Section 3.7.1 of this document for angle tracking deficiencies discovered during function level validation. The extra checks and resets associated with MANAIM and MEMODE could be eliminated, and the model enhanced if the fire-control computer (FCC) and Manual Aimer passed an error signal just as the discriminators do. This would mean the servo response is still used, as it should be, and no checks or additional code would be required internal to the servo routines.

The large azimuth derivative when AZDIFF returns a negative value should be corrected. Tracking circuit descriptions should be added to the *RADGUNS* manual. The discrepancy in the motor/clutch response should be resolved. The developer should examine possible problems that could arise from fixing the matrix size in a routine and also passing it as an argument to the routine. The variable L should also be declared as REAL or used as an INTEGER.

2.27.5.2 Code Quality and Internal Documentation

The overall code quality in MOVANT and the Utility routines is excellent. Internal documentation was accurate and generally adequate, although no units are given in variable definitions and a few variables are not defined. No additional assumptions and limitations, beyond those given in PD^3 (Section 7.1-5), were found.

2.27.5.3 External Documentation

DRAFT